Vignette: Perceptual Compression for Video Storage and Processing Systems

Amrita Mazumdar, Brandon Haynes, Magdalena Balazinska, Luis Ceze, Alvin Cheung, Mark Oskin
Paul G. Allen School of Computer Science & Engineering
University of Washington
Email: {amrita, bhaynes, magda, luisceze, akcheung, oskin}@cs.washington.edu

***Abstract*—Compressed videos constitute 70% of Internet traffic, and video upload growth rates far outpace compute and storage improvement trends. Leveraging perceptual cues like *saliency*, i.e., regions where viewers focus their perceptual attention, can reduce compressed video size while maintaining perceptual quality, but requires significant changes to video codecs and ignores the data management of this perceptual information. This paper describes Vignette, a new compression technique and storage manager for perception-based video compression. Vignette complements off-the-shelf compression software and hardware codec implementations. Vignette's compression technique uses a neural network to predict saliency information used during transcoding, and its storage manager integrates perceptual information into the video storage system to support a perceptual compression feedback loop. Vignette's saliency-based optimizations reduce storage by up to 95% with minimal quality loss, and Vignette videos lead to power savings of 50% on mobile phones during video playback. Our results demonstrate the benefit of embedding information about the human visual system into the architecture of video storage systems.**

***Keywords*—video compression; storage systems**

I. Introduction

Compressed videos constitute 70% of Internet traffic and are stored in hundreds of combinations of codecs, qualities, and bitrates [1]–[3]. New domains of video production—e.g., panoramic (360°), stereoscopic, and light field video for virtual reality (VR)—demand higher frame rates and resolutions, as well as increased dynamic range. Further, the prevalence of mobile devices with high-resolution cameras makes it increasingly easy for humans to capture and share video.

For decades, video codecs have exploited how humans see the world, for example, by devoting increased dynamic range to spatial features (low frequency) or colors (green) we are more likely to observe. One such perceptual cue, *saliency*, describes where in a video frame a user focuses their perceptual attention. As video resolutions grow, e.g., 360° video and 8K VR displays, the salient regions of a video shrink to smaller proportion of the video frame [4]. Video encoders can leverage saliency by concentrating bits in more perceptually interesting visual areas. Prior work, however, does not address the systems challenges of integrating saliency information into large-scale video storage systems [5]–[7]. In this work, we address the challenges of storing and integrating this perceptual data into video storage and processing systems.

DISTRIBUTION STATEMENT A. Approved for public release: distribution is unlimited. Approval ID: XXABW-2018-XXXX.

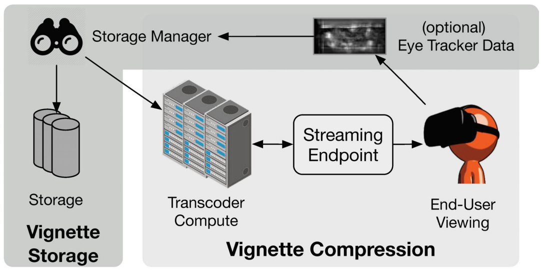


Fig. 1: Vignette provides two features to video storage systems: a perceptual compression algorithm and a storage manager for perceptually compressed videos.

Vignette is a new video storage system integrating perceptual information to reduce video sizes and bitrates. Vignette is designed to serve as a backend for large-scale video services, such as content delivery systems or social media applications. Vignette extends existing, modern codecs to take advantage of the untapped perceptual compression potential of video content, especially high-resolution video served in VR and entertainment settings. We implement Vignette as an extension to LightDB [8], a database management system for video. Our prototype of Vignette demonstrates cost savings to cloud video providers and power savings during mobile video playback. Using a neural network trained to predict content saliency and an off-the-shelf HEVC encoder, our saliency-based compression scheme can reduce bitrate requirements by 80–95%. Our results show that Vignette can reduce whole-system power dissipation by 50% on a mobile phone during video playback. Quantitative evaluation results validate that these savings come at no perceived loss in video quality.

II. Vignette System Design

Vignette consists of two components: Vignette Compression and Vignette Storage; as shown in Figure 1, Vignette Compression is used during the transcoding pipeline, and Vignette Storage manages perceptual information with video data.

Vignette Compression works out-of-the-box with any system that supports HEVC, including hardware accelerators. The algorithm has three high-level steps: (1) generate a saliency map for a given video file, (2) determine the optimal number of rows and columns, which we call a "tile configuration", to spatially partition the video into, and (3) select a per-tile mapping of saliency values to encoder qualities.

Vignette Storage manages perceptual information as simple metadata embedded within videos or maintained in the storage system. This reduces storage complexity for data management and ensures Vignette data is transparent to saliency-unaware video applications such as VLC. The storage manager supports: (1) low-overhead perceptual metadata transmitted alongside video content, (2) storage management policies to trigger one-time perceptual compression, (3) a feedback loop for refining perceptual video compression with cues from user viewing devices, and (4) a heuristic-based search for faster perceptual compression.

A. *Vignette Compression*

Vignette Compression uses off-the-shelf video codec features to encode perceptual information and improve coding efficiency. Our technique takes a video as input, generates a per-frame saliency map for the video, and aggregates the per-frame maps into a single video saliency map. Vignette Compression then transcodes the input video with a tiled encoding, where the quality of each tile corresponds to the saliency of the same tile in the video's saliency map. It uses only the native features of the HEVC codec to ensure compatibility with other video libraries.

1) Automatically Generating Saliency Maps: We use MLNet ([9]) to automatically generate a corresponding saliency map for a video input. Figure 2 shows the saliency map generated for a video frame and how the generated maps capture the visual importance of a given video frame. The process requires decoding the video and processing each frame through the neural network to produce output saliency maps. We accumulate the per-frame saliency maps into a single map by collecting the maximum saliency for each pixel in the frame across the video file. These aggregated saliency values produce a single saliency map of importance across the video.

2) Leveraging Saliency With Tiled Video Encoding: Once a saliency map for each video is produced, we then use it to perceptually encode videos with the tiling feature in HEVC [10]. To produce saliency-based tiled video encoding, we divide a video segment spatially into tiles and then map each tile to a quality setting. The saliency map's value at each tile determines the tile's quality setting. For simplicity and generality, the tiling patterns we use are rectangular tiles with uniform width and height across the video frame. We use the same tile configuration throughout the entire 10-20 second video segment for coding simplicity. We select the number of rows and columns in each a tiling pattern based on either an exhaustive search of all tile configurations or a heuristic-guided search, described in §II-B3.

While tiling is simple and provides coding benefits, a given tile configuration can incur overheads from introducing suboptimal encoding boundaries. A poor tile configuration produces less efficient videos than a standard encoding pass, especially for fast-moving scenes.

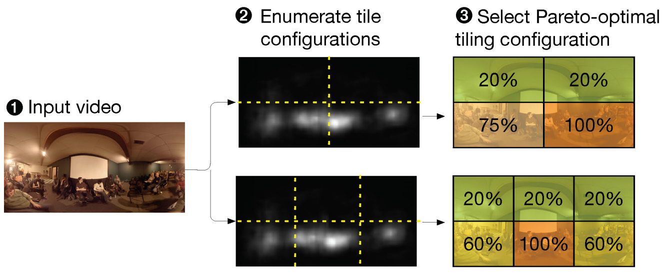

❶ Input video
❷ Enumerate tile configurations
❸ Select Pareto-optimal tiling configuration
20%
20%
75%
100%
20%
20%
20%
60%
100%
60%

Fig. 2: Overview of Vignette Compression algorithm.

We minimize the penalty of adding tile boundaries in areas that would benefit from being encoded together by exhaustively enumerating all tile configurations. We consider only uniform-sized tiles by evaluating across all row-column pairs a video frame allows. In practice, we enumerate tile configurations ranging from 2×2 to 5×10 and 10×5, compress the tiles according to their saliency values, and measure the resulting bitrate and video quality achieved. This exhaustive enumeration takes about 30 minutes per 15-second video to find the best tile configuration with our experimental setup.

3) Mapping Saliency to Video Quality Rates: Each HEVC tile is encoded at a single quality or bitrate setting throughout the video stream, requiring Vignette Compression to select per-tile encoding qualities. We deconstruct saliency maps into per-tile parameters by mapping the highest encoding quality to the maximum saliency value in the tile's saliency map. For evaluation simplicity, we use a perceptually-controlled version of a target bitrate, where the target bitrate either corresponds to the bitrate of the original video or is specified by the API call. The highest-saliency tiles in the video are assigned the target bitrate, and tiles with lower saliency are assigned lower bitrates, with a minimum bitrate of 10% the original video bitrate. As shown in Figure 2, we encode a 0-255 saliency map as discrete bitrates corresponding linearly from a minimum value to the target bitrate or quality, which is the maximum.

B. *Vignette Storage*

Vignette Storage exposes perceptual video compression to applications by providing three features: (1) transparent perceptual metadata, (2) simple storage management policies, and (3) a search algorithm that reduces transcoding cost. A 360° video player, for example, can initialize videos to be oriented in the direction of a high-saliency region it decodes from Vignette metadata, but the videos can also be played traditionally in a standard video player like VLC. Vignette Storage can switch between open and closed-feedback loops for perceptual transcoding; in "open loop" mode, a video is perceptually compressed once based on automatically generated saliency maps, and in "closed loop" mode, perceptually compressed video can be updated based on cues from user-end viewing devices. The heuristic search feature included in Vignette Storage leverages intrinsic video features to enable ∼30× faster perceptual transcoding at near-optimal quality results.

TABLE I: Vignette API

Function	Compression Type	Data required
`transcode`	General	`<IN video, IN CRF/target bitrate, OUT video>`
`vignette_transcode`	Perceptual	`<IN video, (IN CRF/target bitrate,) OUT video, OUT saliency metadata>`
`vignette_squeeze`	Perceptual	`<IN video, IN CRF/target bitrate, OUT video>`
`vignette_update`	Perceptual	`<IN video, IN fixation map, OUT video, OUT saliency metadata>`

1) Saliency Map Metadata: Video storage systems maintain containers of compressed video data that store relevant video features in metadata. Vignette Storage adopts this approach, and injects a small amount ($\sim$100 bytes) of saliency metadata inside each video container. We encode this map as a bitstring that includes fields for the number of rows and columns used for tiled saliency and the saliency weights for each tile. These bitstrings typically range in size from 8–100 bytes. The metadata is included as a saliency `trak`, similar to other metadata atoms in a video container, so that applications with and without perceptual support can decode Vignette videos.

2) Vignette Storage API: The Vignette Storage API defines functions to support open- and closed-loop modes. Table I shows the programming interface for Vignette, which includes three perception-specific operations: `vignette_transcode()`, `vignette_squeeze()`, and `vignette_update()`. Each API operation ingests a video and some required parameters and outputs a video with any generated perceptual metadata encapsulated in the video container.

Transcode Functions. When a new video is uploaded to the storage system, the storage manager triggers the general-purpose `transcode()` function to transcode the video to any specified bitrates and formats for content delivery. This function takes as input a video and target quality parameter, expressed either by CRF or bitrate, and produces a regularly transcoded video.

The `vignette_transcode()` function is the default saliency-based API call. It takes as input a video and an optional quality or bitrate target, and produces both a video and its corresponding generated saliency metadata. When `vignette_transcode` is triggered, Vignette Storage generates new saliency maps, and then compresses the video according to the target quality expressed.

Quality Modulation Functions. As noted in §II-A3, Vignette Compression maps saliency to quality levels for each tile. A `vignette_squeeze()` call will re-compress a video using a specified, reduced bitrate or quality threshold. It takes in a video, target bitrate, and saliency mapping and produces the newly compressed video. For instance, `vignette_squeeze(input.mp4,100k)` transcodes a previously saliency-encoded video from a higher bitrate to a maximum of 100kbps in the most salient regions. The `vignette_squeeze()` function will recompress videos from a higher quality mapping to a lower one, but it will not transcode low-quality videos to a higher-quality mapping to avoid encoding artifacts.

Functions for Updating Perceptual Maps. Vignette Storage also supports a "closed-loop" mode, where saliency maps are updated with new information from eye tracking devices. To invoke this mode, Vignette Storage uses the `vignette_update()` function to ingest and re-process videos with new perceptual information. Similar to how Vignette constructs per-video saliency maps, `vignette_update()` updates the video's saliency map with eye tracker information by executing a weighted average of the original map and the input eye tracker map.

TABLE II: Video datasets used to characterize Vignette.

Type	Benchmark	Description	Bitrate (Mbps)	Size (MB)
Standard	vbench [12]	YouTube dataset	0.53–470	757
	Netflix [11]	Netflix dataset	52–267	1123
VR	VR-360 [13]	4K-360 dataset	10–21	1400
	Blender [14]	UHD / 3D movies	10–147	6817

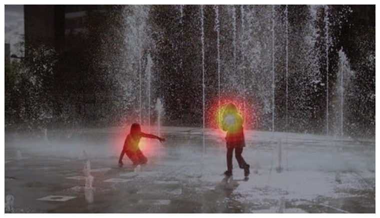
3) Heuristic Search for Tile Configurations: Most of Vignette's computation overhead comes from the exhaustive search over tile configurations for a given video. This exhaustive search is typically performed once, upon video upload, but consumes significant processing time. Vignette Storage contributes a lower cost search algorithm that achieves near-optimal results with a $\sim$30$\times$ performance improvement, for situations where fast saliency-based transcoding is required, e.g., for a newly uploaded video.

Vignette's search technique uses motion vector information from encoded video streams to estimate the size of video tiles. It enumerates tile configurations that group regions of high motion together, and selects a configuration that minimizes the difference in motion vector values across tiles. This heuristic approximates the observation that high-motion areas should not be divided across multiple tiles. Yet, this technique works well because good tile configurations are able to encapsulate redundant motion or frequency information with a single tile, rather than replicate it across tiles. Compared with an exhaustive search, which can transcode a video hundreds of times to empirically produce the optimal tile configuration, our algorithm produces a result $\sim$30$\times$ faster than the exhaustive method and within 1 dB of the best-PSNR result when executed over the videos we use in our evaluation.

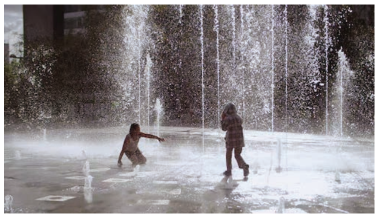
III. Evaluation

We implement Vignette by extending LightDB [8], a database management system for VR videos, and compare against the HEVC implementation included with `FFmpeg`. We measured quality using two visual quality metrics: peak signal-to-noise ratio (PSNR) and eye-weighted PSNR (EWPSNR).

(a) Input video frame from Netflix [11].



(b) Saliency map produced by MLNet [9], overlaid on input.



(c) Perceptually-compressed Vignette video, 85% smaller at iso-quality.

Fig. 3: Example video still, neural network-generated saliency map, and output Vignette perceptually compressed video.

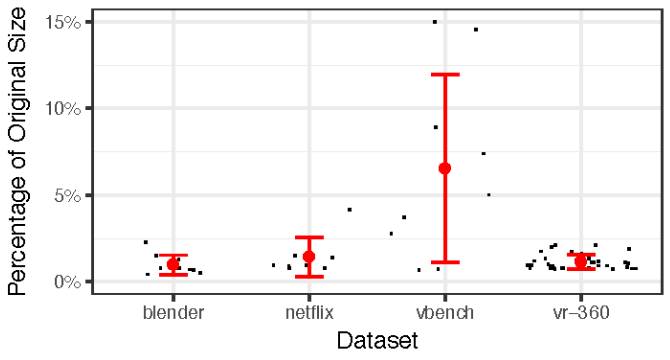


Fig. 4: Aggregate storage savings by dataset. Vignette Compression reduces videos to 1–15% of their original size while maintaining PSNR of 34–39 dB and EWPSNR of 45-51 dB.

A. *Storage and Bandwidth Savings*

We applied Vignette Compression to a corpus of videos, listed in Table II. We transcoded our video library at iso-bitrate in salient regions and decreased bitrate linearly with saliency to a minimum 10% target bitrate in the lowest saliency tiles. Figure 4 shows aggregate storage savings, partitioned by dataset. Overall, we find that Vignette Compression produces videos that are 1–15% of the original size when maintaining the original bitrate in salient regions. These compression savings include the fixed overhead of perceptual metadata, which is <100 B for all videos. Datasets with higher video resolutions (Blender, VR-360) demonstrated the highest compression savings. The vbench dataset, which is algorithmically chosen to have a wide variance in resolution and entropy, exhibits a commensurately large variance in storage reduction. Of the videos with the lowest storage reduction, we find that each tends to have low entropy, large text, or other 2D graphics that are already efficiently encoded.

Table III shows the average reduction in bitrate and resulting quality, measured in PSNR and EWPSNR. Our results show that EWPSNR results are near-lossless for each benchmark dataset, while the PSNR values—which do not take the human visual processing system into account—nonetheless remain acceptable for viewing. Figure 3 highlights a Vignette video frame from the Netflix dataset, with an output PSNR of 36 dB and EWPSNR of 48 dB. Overall, the results indicate that Vignette Compression provides acceptable quality for its compression benefit.

TABLE III: Average bitrate reduction and quality measurements for Vignette Compression by dataset. For PSNR and EWPSNR, $>$ 30 dB is acceptable for viewing, 50 dB+ is lossless.

Benchmark	Bitrate Reduction	PSNR (dB)	Eye-weighted PNSR (dB)
vbench	85.6 %	39	51
Netflix	98.6	34	45
VR-360	98.8	36	45
Blender	98.2	39	49

TABLE IV: Mean processing time per video, evaluated over all videos in our datasets.

	Exhaustive		Heuristic	
Task	Time (s)	%	Time (s)	%
Generate saliency map	1633	49%	1633	95%
Compute tile configuration	1696	50	59	4
Saliency-based transcode	21	1	21	1
Total	3350		1713	

B. *Compute Overhead*

Vignette Compression bears the additional processing overhead of executing a neural network to generate or update saliency maps. Vignette Storage can switch between an exhaustive or more computationally-efficient heuristic tile configuration search to uncover optimal tile configurations for a video. We benchmarked the latency of the combined saliency and transcoding pipeline in two modes: exhaustive, which generates saliency maps per frame and exhaustively evaluates tiling, and heuristic, which uses the heuristic search algorithm to select a tile configuration within 0.25 dB of the best-PSNR choice (§II-B3). Table IV shows generating saliency maps in either mode dominates computation time for Vignette, and that our heuristic search is 33× faster than an exhaustive search. This step, however, is only executed once per video and off the critical path for video streaming workloads.

C. *Analytical Model of Vignette Data Center and Mobile Costs*

We use our evaluation results to model Vignette's system costs at scale for data center storage and end-user mobile power consumption. While these results are a first-order analysis, they suggest the potential benefit of deploying Vignette.

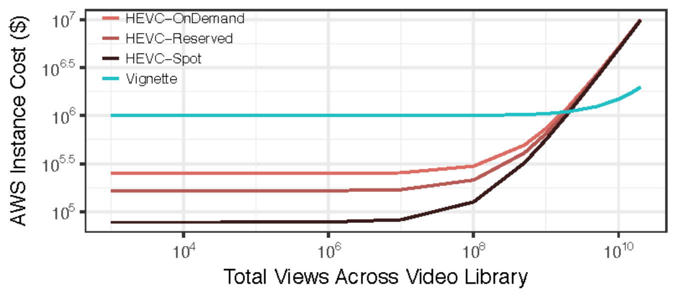


Fig. 5: Estimated AWS costs for deploying Vignette versus traditional video transcoding. Vignette's additional compute cost is amortized after ~2 billion video views over a 1-million video library.

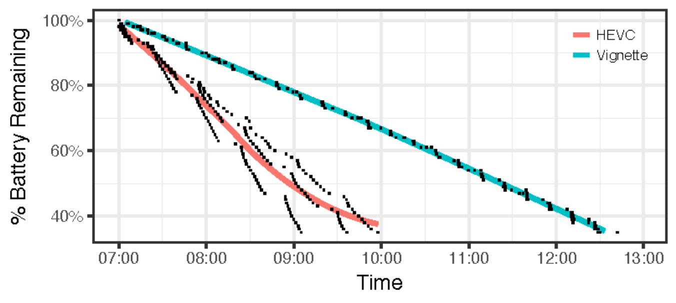


Fig. 6: Time to dissipate a Google Pixel 2 phone battery from 100% to 30% when viewing HEVC and Vignette videos continuously. Vignette videos provide 1.67× longer video playback on mobile phones.

Data center compute, storage, and network costs. We modelled the break-even point for systems that store and deliver video content, using pricing from an Amazon Web Services (AWS) GPU instance, and varied the number of videos transferred to the Internet as a proxy for video views. Larger companies likely use Reserved or Spot Instance offerings, which provide better value for substantial utilization. We assume a video library of 1 million 10-MB videos, encoded at 100 different resolution-bitrate settings to produce ~500 TB of video data. Figure 5 shows how different pricing models produced different savings at small numbers of video library views, but that Vignette becomes cost-effective at large video viewing rates. A system would need to service ~2 billion total views across a million-video library before amortizing Vignette's compute overhead across transmission and storage savings. This number is easily reached by large video services; Facebook reported 8 billion daily views in 2016 [15].

Mobile Power Consumption. To investigate whether Vignette videos can achieve power savings, we profiled power consumption on a Google Pixel 2 phone during video playback of Vignette videos and standard HEVC-encoded videos.

We played our 93-file video library in a loop until the battery charge dissipated from 100% to 30%, conducting three trials each for our HEVC baseline and Vignette videos. Figure 6 shows that Vignette video enabled 1.6× longer video playback time with the same power consumption, or, ~50% better battery life while viewing a fixed number of videos. While hardware decoder implementations are typically proprietary, these results indicate that perceptual compression has benefits for mobile viewers, as well as cloud video infrastructure.

IV. Related Work

Saliency-based compression: Vignette builds on a large body of work in saliency-based compression. Early work improved the accuracy of saliency prediction [5], [16], the speed of computing saliency [7], [17], or coding efficiency [4], [18], [19]; these existing solutions required custom versions of outdated codecs or solving costly optimization problems during each transcoding run. More recently, multimedia and networking research optimized streaming bandwidth requirements for 360° and VR video by decreasing quality outside the VR field-of-view [13], [20], [21]; while similar in spirit to perceptual compression, this only compresses to non-visible regions of a video. Vignette fundamentally differs from other contributions in perceptual compression by introducing a system design that can flexibly use *any* saliency prediction algorithm or video codec, rather than focusing only on accuracy, speed, or efficiency of saliency prediction.

Video streaming and storage systems: The rise of video applications has driven significant recent work in processing and storage systems for video content. Social media services like Facebook or YouTube distribute user-uploaded content from many types of video capture devices to many types of viewing devices, typically serving a small number of popular or livestreamed videos at high quality and low latency, as well as a long tail of less popular videos [22], [23]; this motivated custom media storage and fault-tolerant frameworks for processing videos [1], [12]. Entertainment platforms like Netflix and Amazon Video have smaller video libraries but much more network traffic. These services transcode and store videos for different qualities, network bandwidths, streaming device, and video scene [2], [24], [25]. For both domains, Vignette is a complementary design that solves the challenges of integrating perceptual information with video storage.

V. Future Work

Reducing compute overhead. Vignette's high one-time compression cost is its biggest drawback, but can be improved. Its performance stems from the use of a highly accurate but slow neural network for saliency prediction, which does not yet use a GPU or any modern DL framework optimizations. Further, this expensive compression is run only once, and is easily amortized across many views (§III-C).

Integration with other video system optimizations. We could further improve Vignette by building on other optimizations that work with off-the-shelf video standards. For instance, Vignette's heuristic search algorithm could target more power-efficient tiling configurations with knowledge of open-source video transcoding ASICs [26], [27], or better streaming quality using Fouladi et al.'s codesigned network transport protocol and video codec [28]. Integrating Vignette with these systems could further improve power efficiency during playback, transcoding latency, or archival video storage durability.

VI. Conclusions

Vignette integrates perceptual compression techniques into video storage infrastructure to improve storage capacity and video bitrates while maintaining perceptual quality. Our infrastructure supports a feedback loop of perceptual compression, including updates as an application gathers data from sources such as eye trackers. Our offline compression techniques deliver storage savings of up to 95% with no perceptual quality loss for Vignette videos 50-75% smaller in size. Vignette's design complements the contributions of existing large-scale video storage and processing systems. Video systems can use Vignette to further improve storage capacity or in anticipation of video workloads that produce perceptual information.

Acknowledgment: This work was supported in part by NSF under grant CCF1518703, a generous gift from Google, and by CRISP, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA.

References

[1] Q. Huang, P. Ang, P. Knowles, T. Nykiel, I. Tverdokhlib, A. Yajurvedi, P. Dapolito, IV, X. Yan, M. Bykov, C. Liang, M. Talwar, A. Mathur, S. Kulkarni, M. Burke, and W. Lloyd, "Sve: Distributed video processing at facebook scale," in *SOSP '17*. ACM, 2017. [Online]. Available: http://doi.acm.org/10.1145/3132747.3132775

[2] I. Katsavounidis, "Dynamic optimizer – a perceptual video encoding optimization framework," https://medium.com/netflix-techblog/dynamic-optimizer-a-perceptual-video-encoding-optimization-framework-e19f1e3a277f, Tech. Rep., 2018, accessed: 2018-06-08.

[3] Cisco, "Cisco visual networking index: Forecast and methodology, 20082013," Tech. Rep., 2008, accessed: 2018-06-07.

[4] V. Sitzmann, A. Serrano, A. Pavel, M. Agrawala, D. Gutierrez, B. Masia, and G. Wetzstein, "Saliency in VR: How do people explore virtual environments?" *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 4, April 2018.

[5] Z. Li, S. Qin, and L. Itti, "Visual attention guided bit allocation in video compression," *Image and Vision Computing*, vol. 29, no. 1, 2011.

[6] H. Hadizadeh and I. V. Baji, "Saliency-aware video compression," *IEEE Transactions on Image Processing*, vol. 23, no. 1, Jan 2014.

[7] C. Guo and L. Zhang, "A novel multiresolution spatiotemporal saliency detection model and its applications in image and video compression," *IEEE Transactions on Image Processing*, vol. 19, no. 1, Jan 2010.

[8] B. Haynes, A. Mazumdar, A. Alaghi, M. Balazinska, L. Ceze, and A. Cheung, "LightDB:a DBMS for virtual reality," *Proc. VLDB Endow.*, vol. 11, no. 10, 2018.

[9] M. Cornia, L. Baraldi, G. Serra, and R. Cucchiara, "A Deep Multi-Level Network for Saliency Prediction," in *International Conference on Pattern Recognition (ICPR)*, 2016.

[10] G. J. Sullivan, J. Ohm, W. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, 2012.

[11] Z. Li, A. Aaron, I. Katsavounidis, A. Moorthy, and M. Manohara, "Toward a practical perceptual video quality metric," http://techblog.netflix.com/2016/06/toward-practical-perceptual-video.html, Tech. Rep., 2016, accessed: 2018-06-08.

[12] A. Lottarini, A. Ramirez, J. Coburn, M. A. Kim, P. Ranganathan, D. Stodolsky, and M. Wachsler, "Vbench: Benchmarking video transcoding in the cloud," in *ASPLOS '18*. ACM, 2018. [Online]. Available: http://doi.acm.org/10.1145/3173162.3173207

[13] W.-C. Lo, C.-L. Fan, J. Lee, C.-Y. Huang, K.-T. Chen, and C.-H. Hsu, "360&deg; video viewing dataset in head-mounted virtual reality," in *MMSys'17*. ACM, 2017. [Online]. Available: http://doi.acm.org/10.1145/3083187.3083219

[14] B. Foundation, "Blender open projects," https://www.blender.org/about/projects/, Tech. Rep., 2002, accessed: 2018-07-11.

[15] Mediakix, "The facebook video statistics everyone needs to know," http://mediakix.com/2016/08/facebook-video-statistics-everyone-needs-know, 2016.

[16] J. S. Lee and T. Ebrahimi, "Perceptual video compression: A survey," *IEEE Journal of Selected Topics in Signal Processing*, vol. 6, no. 6, Oct 2012.

[17] R. Gupta, M. T. Khanna, and S. Chaudhury, "Visual saliency guided video compression algorithm," *Signal Processing: Image Communication*, vol. 28, no. 9, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0923596513000982

[18] F. Zund, Y. Pritch, A. Sorkine-Hornung, S. Mangold, and T. Gross, "Content-aware compression using saliency-driven image retargeting," in *Image Processing (ICIP), 2013 20th IEEE International Conference on*. IEEE, 2013.

[19] V. Lyudvichenko, M. Erofeev, Y. Gitman, and D. Vatolin, "A semi-automatic saliency model and its application to video compression," in *2017 13th IEEE International Conference on Intelligent Computer Communication and Processing (ICCP)*, Sept 2017, pp. 403–410.

[20] C.-L. Fan, J. Lee, W.-C. Lo, C.-Y. Huang, K.-T. Chen, and C.-H. Hsu, "Fixation prediction for 360&deg; video streaming in head-mounted virtual reality," in *NOSSDAV'17*. ACM, 2017.

[21] B. Haynes, A. Minyaylov, M. Balazinska, L. Ceze, and A. Cheung, "Visualcloud demonstration: A dbms for virtual reality," in *SIGMOD '17*. ACM, 2017. [Online]. Available: http://doi.acm.org/10.1145/3035918.3058734

[22] "Facebook live — live video streaming," https://live.fb.com/, 2018.

[23] L. Tang, Q. Huang, A. Puntambekar, Y. Vigfusson, W. Lloyd, and K. Li, "Popularity prediction of facebook videos for higher quality streaming," in *ATC '17*. USENIX Association, 2017. [Online]. Available: https://www.usenix.org/conference/atc17/technical-sessions/presentation/tang

[24] A. Aaron, Z. Li, M. Manohara, J. D. Cock, and D. Ronca, "Per-title encode optimization," https://medium.com/netflix-techblog/per-title-encode-optimization-7e99442b62a2, Tech. Rep., 2015, accessed: 2018-06-08.

[25] M. Manohara, A. Moorthy, J. D. Cock, I. Katsavounidis, and A. Aaron, "Optimized shot-based encodes: Now streaming!" https://medium.com/netflix-techblog/optimized-shot-based-encodes-now-streaming-4b9464204830, Tech. Rep., 2018, accessed: 2018-06-08.

[26] I. Magaki, M. Khazraee, L. V. Gutierrez, and M. B. Taylor, "Asic clouds: Specializing the datacenter," in *ISCA '16*. Piscataway, NJ, USA: IEEE Press, 2016. [Online]. Available: https://doi.org/10.1109/ISCA.2016.25

[27] H. Zhang, P. V. Rengasamy, S. Zhao, N. C. Nachiappan, A. Sivasubramaniam, M. T. Kandemir, R. Iyer, and C. R. Das, "Race-to-sleep + content caching + display caching: A recipe for energy-efficient video streaming on handhelds," in *MICRO-50 '17*. ACM, 2017. [Online]. Available: http://doi.acm.org/10.1145/3123939.3123948

[28] S. Fouladi, J. Emmons, E. Orbay, C. Wu, R. S. Wahby, and K. Winstein, "Salsify: Low-latency network video through tighter integration between a video codec and a transport protocol," in *NSDI '18*. Renton, WA: USENIX Association, 2018. [Online]. Available: https://www.usenix.org/conference/nsdi18/presentation/fouladi